МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до самостійної роботи
«Створення та налаштування програм у візуальному середовищі Delphi 2009»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Затверджено редакційно-видавничою
радою університету,
протокол № 2 від 01.12.10.

Харків НТУ «ХПІ» 2011

Методичні вказівки до самостійної роботи «Створення та налаштування програм у візуальному середовищі Delphi 2009» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика (спеціалізація «Соціальна інформатика») / Уклад. М. І. Безменов. – Х. : НТУ «ХПІ», 2011. – 41 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

ВСТУП

Отримання навичок програмування у сучасних середовищах програмування неможливе без самостіної роботи за комп'ютером.

Метою даної самостійної роботи є отримання уявлення про структуру програми у Delphi, освоєння середовища розробки Delphi 2009 та отримання початкових навичок у проектуванні Windows-застосунків з використанням компонентів Delphi.

1. ЗАГАЛЬНИЙ ОПИС ПРОГРАМИ, НАПИСАНОЇ МОВОЮ DELPHI

1.1. Поняття класу загальні відомості

Об'єктно-орієнтоване програмування – це методологія програмування, заснована на організації програми у вигляді сукупності ***об'єктів***, кожний з яких є представником певного ***класу***, а класи утворюють ієрархію спадкування.

Центральним елементом абстракції при цьому є об'єкт, що у програмі являє собою модель «предмета» реального світу.

Об'єкт насамперед характеризується станом, що визначається набором характеристик та їх поточних значень. Ці характеристики називають ***полями***. Не всі характеристики об'єкта видні ззовні, – деякі з них сховані. Говорять, що такий стан інкапсульовановано в об'єкт.

Ще одним поняттям, що пов'язане з об'єктом, є взаємодія його зі своїм оточенням (іншими об'єктами). Маніпуляції (дії) з об'єктом викликають деяку його реакцію. Звичайно операції над об'єктами називають ***методами***. Вони реалізуються особливим чином оголошеними та оформленими підпрограмами.

Особливим видом полів є ***властивості***, які, подібно звичайним полям, визначають характеристики екземплярів класів. Але в той час як поле – це просто ділянка пам'яті, вміст якої може змінюватися, властивість зв'язує певні дії із читанням даних, записаних у ній, або їх зміною. При цьому властивості забезпечують керування доступом до полів.

Клас – це множина об'єктів, що мають однакову структуру. Конкретний об'єкт є представником класу, або екземпляром класу.

Структура опису класу наступна:

```
class ім'я_класу
   поля
   методи
   властивості
end;
```

Видимість компонентів класу визначається службовими словами **private** (власний), **public** (загальнодоступний), **protected** (захищений). Ці службові слова відкривають окремі розділи у визначення класу і є специфікаторами доступу. Поява будь-якого специфікатора доступу в тексті визначення класу означає, що до кінця визначення або до іншого специфікатора доступу всі компоненти класу мають зазначений статус. Специфікатор **private** означає, що поля, властивості та методи, розміщені під ним, доступні тільки методам даного класу. Це так званий закритий доступ.

Специфікатор **public** (відкритий доступ) означає, що розміщені під ним елементи доступні як даному класу, так і методам інших класів і взагалі будь-яким підпрограмам через представника класу.

Специфікатор захищеного доступу **protected** означає, що поля і методи, розміщені під ним, доступні не тільки в даному класі, але й для методів класів, похідних від нього.

У Delphi класи можуть мати елементи розміщені в секції зі специфікатором доступу **published** (опубліковані). Оголошення елементів класу в цьому розділі еквівалентно оголошенню в розділі **public**, але додатково опубліковані властивості та методи доступні в режимі проектування через інспектор об'єктів. Відсутність спціфікатора доступу є еквівалентною наявності спеціфікатора **published**.

Особливістю Delphi-програм є активне використання компонентів. Звичайно під *компонентом* розуміють деякий елемент, що має свою функціональність і може бути розміщений програмістом на формі на етапі проектування програми. У певному розумінні в працюючій програмі компоненти являють собою об'єкти «реального світу»: їх можна переміщати за допомогою мишки або клавіш, вони реагують на натискання клавіш і кнопок мишки. Використання компонентів забезпечує як наочність і зручність роботи при проектуванні програми (насамперед її інтерфейсної частини), так і зручність

її експлуатації. Основна різниця між компонентами та об’єктами інших класів полягає в тому, що компонентами можна маніпулювати на формі, а об’єктами ні.

Компоненти в Delphi програмуються за допомогою властивостей, методів і подій. З погляду користувача компонента ***подія*** – це індикатор (ознака) виникнення якої-небудь ситуації, на яку повинен відреагувати компонент. Реакція на події реалізується так званими ***опрацьовувачами подій***, які є методами, що найчастіше за все належать класу тієї форми, на якій розташований компонент. Така методика передачі об’єктом частини своїх функцій іншому об’єкту називається ***делегуванням***.

Якщо виконується яка-небудь дія з компонентом (переміщення або клік мишкою, натискання клавіші й т. д.), компонент генерує подію, результатом чого може бути генерування системою інших подій. Технічно більшість подій в Delphi запускається при одержанні компонентом повідомлення від операційної системи.

Формально подія – це властивість, яка має тип «вказівник методу». У неї записано адресу завантаження опрацьовувача події. Будучи оголошуваними у секції **`published`**, події відображаються в Інспекторі Об'єктів, причому всі властивості, що є вказівниками на методи, Delphi відображає на вкладці Events (Події), а всі інші властивості – на вкладці Properties (Властивості).

За згодою події залежно від змістового навантаження йменуються ідентифікаторами, які мають один із префіксів `On`, `After`, `Before` (наприклад, `OnClick` або `OnChange`).

Методи, відповідальні за виклик події компонента, називаються ***методами диспетчеризації подій***. За згодою їх іменують або ім’ям події без префікса, або після відкидання префікса додають на початок імені методу префікс `Do` (наприклад, методом диспетчеризації події OnClick класу `TControl` є метод `Click`, у той час як метод диспетчеризації події OnCreate класу `TForm` названий `DoCreate`, оскільки ім’я `Create` дане конструкторові). У деяких випадках ім’я методу диспетчеризації починають із одного з префіксів `After` або `Before`.

При створенні заданих за умовчанням імен опрацьовувачів подій для компонентів Delphi як префікс використовує ім’я компонента.

Для створення опрацьовувачів подій на етапі проектування застосунка може бути застосований Object Inspector (Інспектор Об'єктів). Насамперед необ-

хідно перейти в ньому на вкладку Events (Події) і кліком мишкою в лівому стовпчику вибрати потрібну подію. У результаті в правому стовпчику вкладки відкриється віконце, у правій частині якого розташована кнопка . Клік мишкою над цією кнопкою приводить до відкриття списку вибору. У цьому списку можна вибрати ім'я опрацьовувача події, що вже визначений у проекті і може бути використаний як опрацьовувач даної події. Якщо вміст списку, що буде відкритий, не задовольняє програміста (що буває досить часто) або такий список відсутній, то можна набрати ім'я створюваного опрацьовувача події, після чого перейти у вікно коду з метою безпосереднього написання тексту опрацьовувача події. Перехід у вікно коду може бути виконаний одним з наступних способів:

- натисканням клавіші Enter після набору імені опрацьовувача події;
- подвійним кліком мишки над набраним ім'ям опрацьовувача події;
- подвійним кліком мишкою над пустим віконцем праворуч від імені події на вкладці Events (Події).

При цьому буде автоматично змінений код (доданий відповідний новий метод в опис форми і вставлена заготівка для реалізації цього методу). Курсор розташується у виконуваній частині вставленої заготівки методу, після чого можна перейти до безпосереднього написання програмного коду, що реалізує опрацьовувач події.

Якщо клікнути мишкою у дизайнері форми над зображенням компонента на формі під час її проектування, то здійснюється вставка заготовки опрацьовувача події за умовчанням. Наприклад, для кнопки такою подією є подія OnClick, для форми – подія OnCreate.

1.2. Проект

Мінімальний код, що може нормально функціонувати під керуванням операційної системи, створюється автоматично відразу ж після вибору команди File ► New ► VCL Forms Application – Delphi (Файл ► Створити ► Застосунок з VCL формами).

Якщо говорити про текст програми, то як такий можна розглядати так званий файл проекту, що має розширення .DPR. Саме цей файл оброблюється компілятором, будучи головною програмною одиницею, до якої підключаються одна або кілька інших програмних одиниць, що називаються модулями та зберігаються у файлах з розширенням .PAS.

Програма містить три основні частини:

- заголовок програми;
- розділ описів;
- тіло програми.

Заголовок програми починається зі слова **program**, після якого зазначається ім'я програми та ставиться крапка з комою.

За заголовком програми розташовується ***розділ описів***, у якому оголошуються так звані ідентифікатори, які позначають різні елементи програми (модулі, константи, типи, змінні, процедури, функції).

Власне ***тіло програми*** починається зі слова **begin** і завершується словом **end** з крапкою, яке інакше називається ***термінатором***. Тіло програми складається з декількох операторів мови Delphi. У кожному операторі реалізується певна дія, наприклад, змінювання значення змінної, аналіз результату обчислення, звертання до підпрограми тощо.

Стандартний текст проекту для функціонування форми у разі проектування застосунка у середовищі програмування Delphi 2009 має такий вигляд:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Зазвичай (якщо не проводилося переналаштування) у вікні коду автоматично виділяються жирним шрифтом так звані зарезервовані слова і курсивом – коментарі.

Розглянемо текст проекту.

У першому рядку розташований заголовок програми з автоматично наданим ім'ям `Project1`. Це ім'я може бути змінене програмістом відповідно до змісту задачі.

Зарезервоване слово **uses** (використовується) служить для відкриття розділу опису модулів. У цьому розділі підключаються стандартний модуль `Forms` і створений модуль форми `Unit1`, невідомий Delphi, і тому зазначене ім'я файлу, який містить текст цього модуля (`Unit1.pas`). Крім того, у вигляді коментарю Delphi повідомляє ім'я описаної в модулі форми (стандартне написання коментарів – курсивне).

Для забезпечення нормального функціонування програми до неї повинен бути підключений файл ресурсів Windows. Це виконується на етапі компонування програми, і вказівкою на необхідність підключення файлу ресурсів буде рядок `{$R *.res}`, що є однією з так званих директив компілятору. У разі відсутності раніше створеного фаулу ресурсів він створюється автоматично при компіляції програми, дістаючи ім'я, що збігається з ім'ям проекту, і розширення .RES.

При виконанні програми завжди автоматично створюється спеціальний об'єкт, який має ім'я `Application` і є об'єктом-програмою. Цей об'єкт акумулює в собі дані та методи їх обробки, необхідні для нормального функціонування Windows-програми.

Стандартне тіло проекту у Delphi 2009 містить чотири оператори, що реалізують звертання до трьох методів об'єкта `Application`. Перший з цих методів, а саме,

```
Application.Initialize,
```

здійснює виклик спеціальної підпрограми, ім'я якої записане в системній змінній `InitProc`.

Оператор

```
Application.MainFormOnTaskbar := True
```

вказує, що в панелі задач буде відбиватися рядок заголовка головної форми разом з піктограмою.

Якщо видалити цей оператор або записати в ньому замість значення `True` значення `False`, в панелі задач буде виводитись ім'я проекту (наприклад, Project1) або рядок, записаний у властивість `Title` об'єкта `Application`. Так, якщо замість оператора

```
Application.MainFormOnTaskbar := True
```

записати оператори

```
Application.MainFormOnTaskbar := False;
Application.Title:='Моя перша програма';
```

в панелі задач буде відбиватися рядок Моя перша програма.

Оператор

```
Application.CreateForm(TForm1, Form1)
```

служить для створення та показу на екрані вікна головної форми. Закриття цього вікна припиняє виконання програми.

Метод `Application.Run` забезпечує одержання та опрацювання повідомлень, які в ході виконання програми надходять від операційної системи, будучи сигналами про дії користувача або про роботу різного роду пристроїв.

Файл проекту висвітлюється у вікні коду або на запит програміста, або з появою деяких помилок у ході виконання програми. Звичайно у вікні коду він не відображається і змінюється тільки достаньо кваліфікованим програмістом, і то досить рідко.

1.3. Модуль форми

Модулі – це програмні одиниці, що служать для розміщення окремих частин програм. Будь-який модуль (у тому числі і модуль форми) має таку структуру:

- заголовок, який відкривається зарезервованим словом **`unit`**;
- секція інтерфейсних оголошень, що відкривається словом **`interface`**;
- секція реалізацій, що відкривається словом **`implementation`**;
- термінатор **`end`** з крапкою.

Після секції реалізації (перед термінатором) **можуть** бути розміщені ініціююча та (або) завершальна секції, що починаються відповідно зі службових слів **`initialization`** та **`finalization`**.

Заголовок модуля, як і заголовок проекту, завершується крапкою з комою.

Мінімальний синтаксично правильний варіант модуля має такий вигляд:

```
unit Unitl;

interface
```

```
// Секція інтерфейсних оголошень
implementation
// Секція реалізацій
end.
```

У секції інтерфейсних оголошень вміщується опис програмних елементів (констант, типів, процедур і т. д.), які будуть відомі («видні») іншим програмним модулям, що містять посилання на даний модуль. У секції ж реалізацій розкривається механізм роботи цих елементів.

Розділення модуля на дві секції забезпечує зручний механізм доступу до його ресурсів. Щоб отримати доступ до ресурсів модуля достатньо знати тільки особливості його інтерфейсної частини, що містить оголошення елементів. Деталі ж їхньої реалізації (йдеться про процедури і функції) залишаються прихованими.

Модулем форми є модуль, до якого входять елементи програмного коду, які забезпечують створення та функціонування форми та тих компонентів, що розміщуються на ній.

На вкладці Unit вікна коду у випадку стандартної форми міститься код, сформований Delphi, що в обов'язковому порядку повинен бути змінений програмістом (рис. 1.1).

В інтерфейсній секції наведеного програмного коду підключені стандартні модулі, а також описані один тип (клас `TForm1`) та один об'єкт (змінна `Form1`).

Об'єкти, що використовуються в програмі, створюються за деякими зразками, які називаються ***типами***. Перед оголошенням типу вказується службове слово **`type`** (тип), що повідомляє компілятор про початок розділу опису типів. Одним з різновидів типів є клас (на рис. 1.1 він має ім'я `TForm1`).

Все потрібне для створення та функціонування порожнього вікна програми реалізовано в стандартному класі `TForm`. Будь-який клас може бути удосконалений (розвинений). Однак для забезпечення цього програмісти звичайно не змінюють вже існуючий клас, а породжують від нього новий клас, додаючи йому додаткову функціональність і зберігаючи всі можливості батьківського класу.

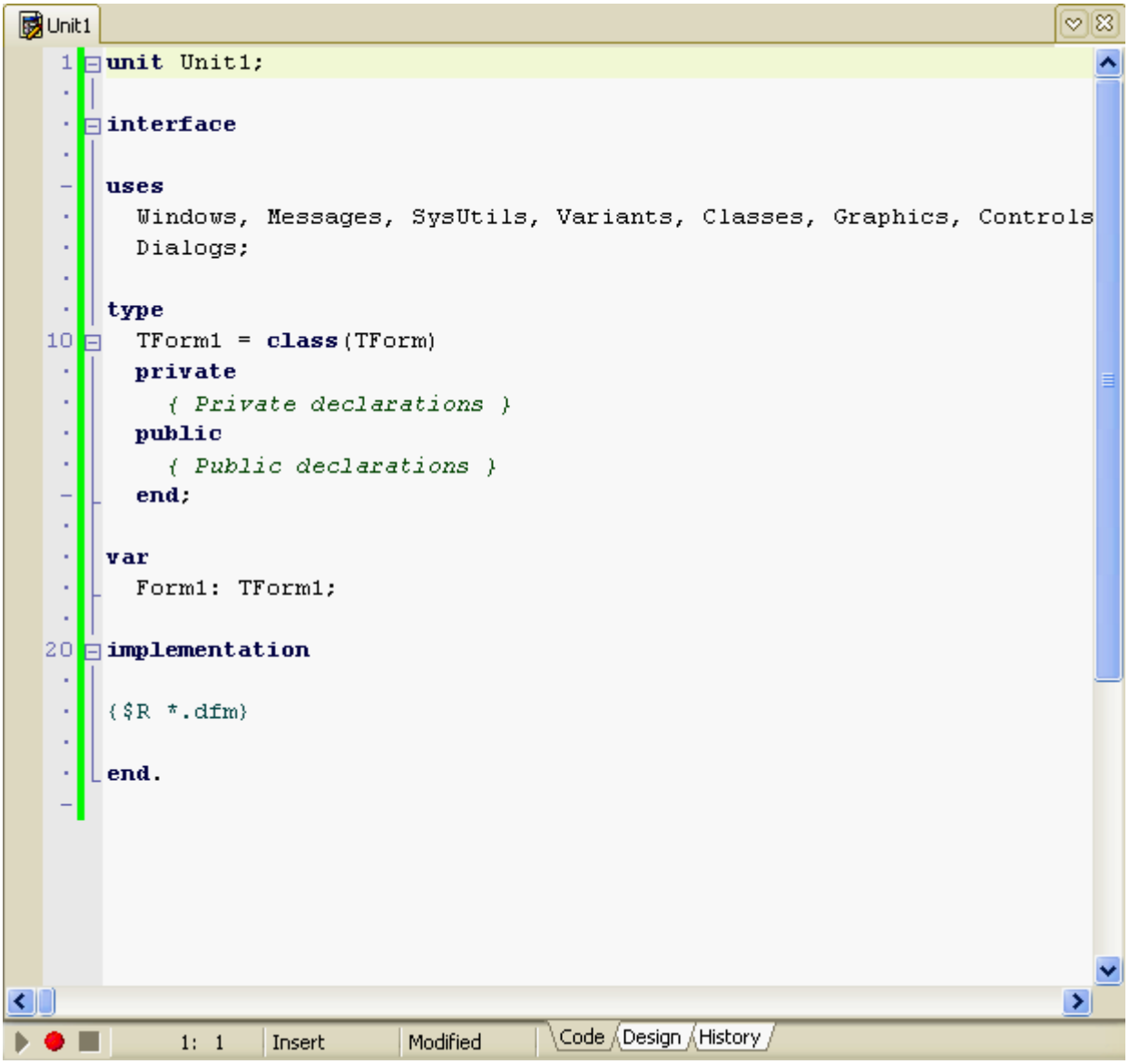


Рис. 1.1 – Початковий вигляд вікна коду

У наведеному тексті відбито, що клас `TForm1` *породжений* від стандартного класу `TForm`. Про це свідчить рядок

```
TForm1 = class(TForm)
```

У цьому випадку, оскільки поки що йдеться про порожню форму, додаткові можливості у класу `TForm1` відсутні.

Важливими елементами програми є ***змінні***. Для того щоб їх можна було використовувати, вони попередньо повинні бути описані. Оскільки об'єкт `Form1` є змінною, він описаний в інтерфейсній частині модуля і буде відомим у проекті, у якому здійснене підключення даного модуля. Перед

оголошенням об’єкта вказано зарезервоване слово **`var`** (від англ. *variables* – змінні), що інформує компілятор про початок розділу опису змінних.

Якщо на етапі проектування програміст розміщає на формі новий компонент, то в текст модуля автоматично вставляється опис цього компонента та за додатковими командами створюються заготівки опрацьовувачів подій, на які реагує цей компонент. Справа програміста – наповнити опрацьовувачі подій конкретним змістом у вигляді операторів. Програміст може оголошувати нові змінні, типи, константи і т. д., однак видаляти та змінювати рядки, вставлені автоматично, не рекомендується. Наприклад, якщо у тексті модуля є опрацьовувач деякої події і треба його видалити, то необхідно знищити тільки рядки, що вставлялися програмістом. При компіляції ж буде видалений увесь текст, що вставлявся автоматично.

2. РОБОТА У ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБЛЮВАЧА DELPHI 2009

2.1. Початкові дії

Інтегроване середовище розроблювача (ICP) Delphi 2009 візуально реалізується декількома одночасно відкритими вікнами, розміщеними у головному вікні Delphi. При завантаженні Delphi 2009 у центральній частині ICP Delphi міститься вікно зі сторінкою вітання (Welcome Page) CodeGear RAD Studio. При подальшій роботі це вікно залишається доступним (якщо його не закрити спеціально) із забезпеченням доступу до нього кліком мишкою над його закладкою. У вигляді окремих сторінок у тому ж самому вікні, у якому розташовується сторінка вітання, містяться вікно дизайнера форми та сторінки з кодами модулів. Праворуч угорі розташовуються дві кнопки, перша з яких () служить для виклику випадного меню, яке забезпечує перехід до тієї або іншої сторінки (перехід можна виконати і кліком мишкою над відповідною закладкою у верхній частині вікна), а друга () – для закриття активної сторінки. Зокрема, за допомогою сторінки вітання забезпечується швидкий доступ до недавніх проектів, створення нового і відкриття існуючого проекту. Якщо сторінку вітання закрили, то повторно її відкривають командою View ► Welcome Page.

Після завантаження середовища потрібно здійснити перехід до розробки нового застосунка. Порядок дій при цьому може бути таким:

1. Вибрати у головному меню опцію File, наслідком чого буде розкриття підменю, першою опцією якого є опція New.
2. У розкритому меню вибрати опцію New, в результаті чого відкривається підменю, однією з опцій якого є опція Other... (Інший...).
3. Вибрати опцію Other..., що приведе до відкриття вікна New Items (див. рис. 2.1).
4. У вікні New Items вибрати пункт VCL Forms Application, наслідком чого буде відкриття вікон для проектування застосунка (рис. 2.2).

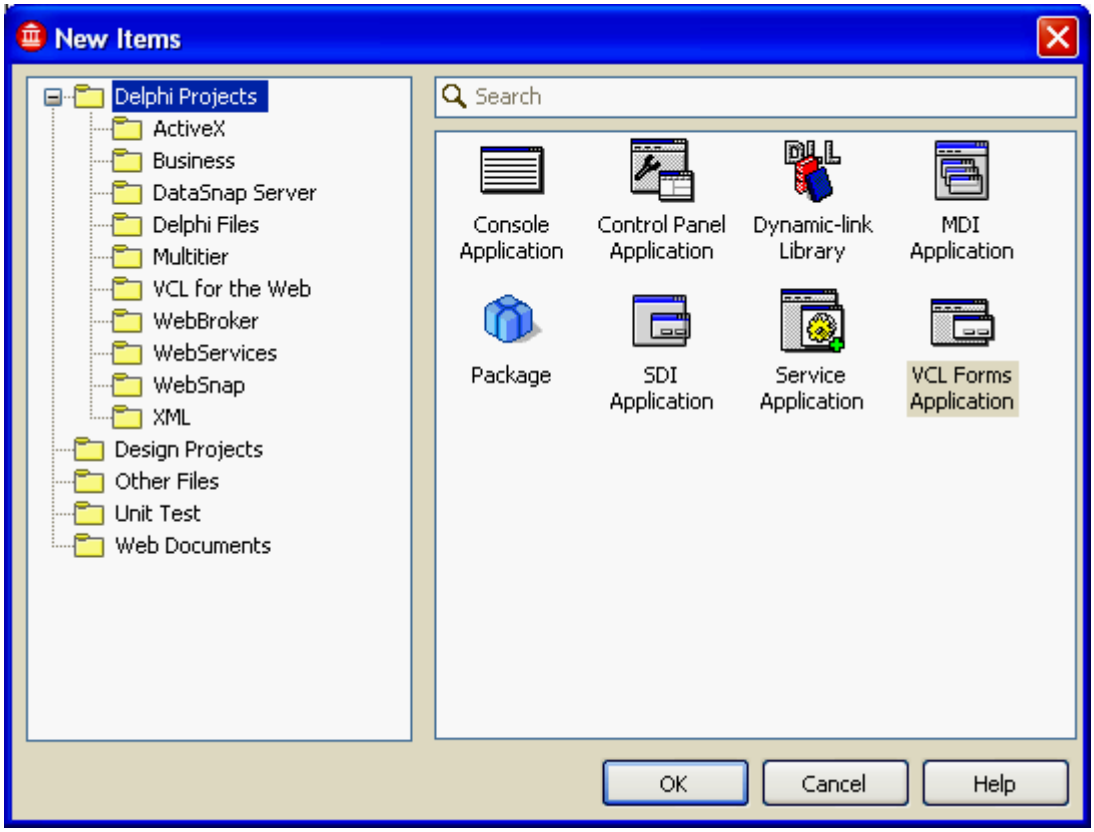


Рис. 2.1 – Вікно вибору виду застосунка, що проектуватиметься

Послідовність дій пунктів 1–4 можна позначити так:

File ►New ►Other... ► VCL Forms Application

Примітка. Можна одразу увійти у вікно New Items, що відкривається за командою File ►New. Для цього треба клікнути мишкою над інструментальною кнопкою (New Items).

Відзначимо, що одразу після виконання п. 4 описаної вище послідовності дій буде автоматично сформований проект, який вже може бути виконаний. Цей проект забезпечує функціонування порожньої форми.

Серед зазначених на рис. 2.2 вікон тільки головне вікно обов’язково є присутнім на екрані. Інші вікна можуть бути закриті, якщо вони не потрібні проектувальнику застосунка.

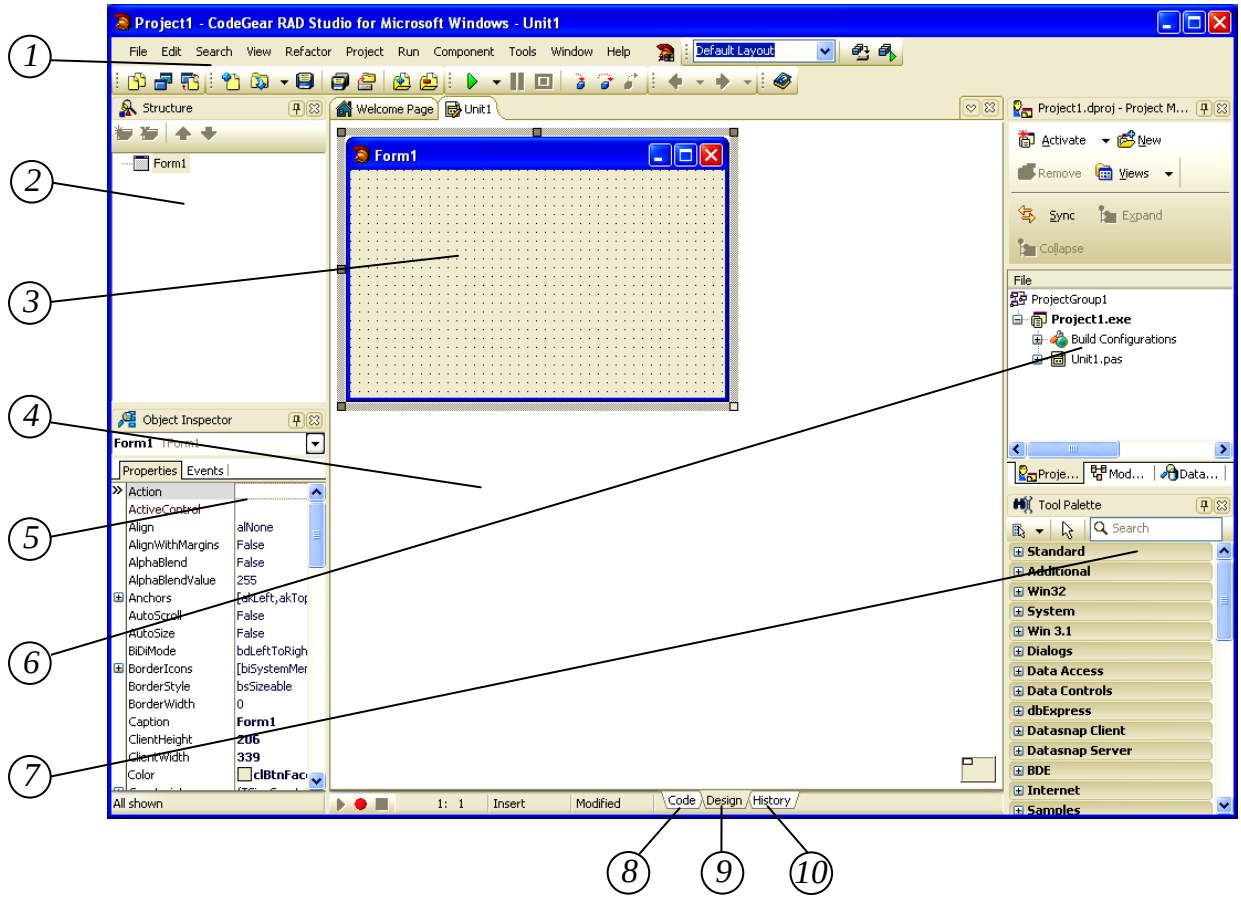


Рис. 2.2 – Вікна ICP Delphi 2009 на початку роботи над новим проектом: *1* – головне вікно; *2* – вікно структури; *3* – вікно форми; *4* – вікно дизайнера форми; *5* – вікно інспектора об’єктів; *6* – вікно менеджера проектів; *7* – вікно палітри інструменту; *8* – закладка вікна коду; *9* – закладка вікна дизайнера форми; *10* – закладка вікна менеджера історії

2.2. Початкові відомості про середовище розроблювача Delphi 2009

2.2.1. Головне вікно

Це вікно містить у собі всі засоби з керування проектом (рис. 2.3), а саме, головне меню та набір інструментальних кнопок.

Усі елементи головного вікна, крім головного меню, згруповані на окремих панелях, які мають спеціальні вішки, призначені для переміщення

цих панелей при їхньому захоплюванні мишкою і перетаскуванні при натиснутій її лівій кнопці. Будь-яку з цих панелей можна прибрати з екрана.

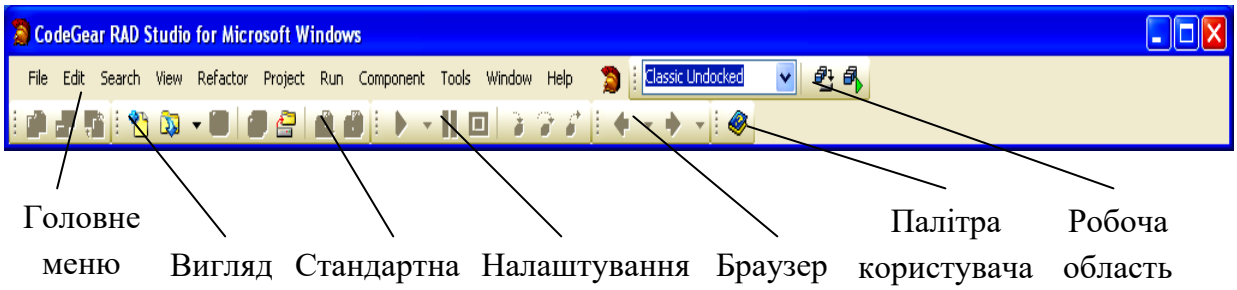


Рис. 2.3 – Основні панелі головного вікна ICP Delphi

За допомогою головного меню здійснюється керування можливостями ICP. Швидкий доступ до багатьох команд головного меню забезпечують так звані інструментальні кнопки, згруповані за функціональним призначенням на окремих панелях головного вікна. Панель Стандартна містить інструментальні кнопки, що забезпечують дублювання тих дій, які можна виконати, звернувшись до пунктів File (Файл) і Project (Проект) головного меню. Панель Вигляд поєднує кнопки, що дублюють деякі підпункти пункту View (Вигляд) головного меню, а панель Налаштування полегшує керування процесом налаштування та виконання програми, частково дублюючи підпункти пункту Run (Виконати) головного меню. За допомогою кнопок, що розміщені на панелі Браузер, можна переходити до сторінок, з якими здійснювалася робота при звертанні до Internet-ресурсів. Панель Робоча область дозволяє здійснювати керування виглядом робочої області ICP Delphi (а саме, за умовчанням, для режиму налаштування, класичний), а також збереження поточного її вигляду (за допомогою інструментальної кнопки ). Панель Палітра користувача за умовчанням вміщує тільки одну інструментальну кнопку, що забезпечує доступ до довідкової системи Delphi.

У верхній частині головного вікна міститься також кнопка  , яка забезпечує відкриття вікна з довідковими даними про версію Delphi.

2.2.2. Вікно дизайнера форми

На початку роботи над новим проектом вікно дизайнера форми у вигляді окремої сторінки із закладкою Design розташовується над сторінкою вітання, за умовчанням будучи відкритим. У правому нижньому куті сторін-

ки дизайнера форми виводиться невеликий затінений прямокутник, який умовно відображає екран при роботі застосунка (рис. 2.2). На зображенні екрана відображається місце розташування і розміри форми відразу ж після запуску застосунка. При зміні розміру або місця розташування форми на етапі її конструювання автоматично змінюються розміри і місце розташування її зображення на зображенні екрана. Захопивши мишкою зображення форми на зображенні екрана, можна виконати перетаскування її з автоматичною зміною координат. Клік мишкою над зображенням екрана забезпечує збільшення цього зображення відносно початкових розмірів, причому в цьому випадку можливе захоплення мишкою лівого верхнього кута зображення екрана з метою його пропорційного розтягування або зменшення. За наявності у проекті декількох форм дизайнер форми містить кілька сторінок, закладки яких розміщуються у верхній його частині. Кожна з форм у цьому випадку відображується на зображенні екрана, причому зображення форми, з якою здійснює роботу дизайнер форми, залите білим кольором, а зображення інших форм затінені (див. рис. 2.4 для проекту з двома формами).

У лівому верхньому куті вікна дизайнера форми на етапі проектування розташовується вікно форми. Розміри форми можуть змінюватися за допомогою мишки, для чого слід захопити праву або нижню її межі або правий нижній кут. З метою полегшення вирівнювання компонентів дизайнер форми показує напрямні лінії, які висвітлюються при перетаскуванні компонентів за допомогою мишки.

Вікно форми являє собою проект вікна створюваної програми. Спочатку воно містить кнопки виклику системного меню, розгортання, згортання і закриття вікна, рядок заголовка і габаритну рамку (ці елементи є стандартними інтерфейсними елементами Windows), а також порожню робочу область, звичайно заповнену крапками координатної сітки, що служить для полегшення позиціювання розташовуваних на формі компонентів і відображається тільки на етапі конструювання програми.

За допомогою команди Tools ► Options... ► Environment Options ► VCL Designer (Сервіс ► Опції...► Опції Середовища ► Дизайнер VCL) можна викликати вікно налаштувань дизайнера бібліотеки візуальних компонентів (Visual Component Library, VCL) і, знявши прапорець Display Grid (Показувати Сітку), прибрати координатну сітку з форми.

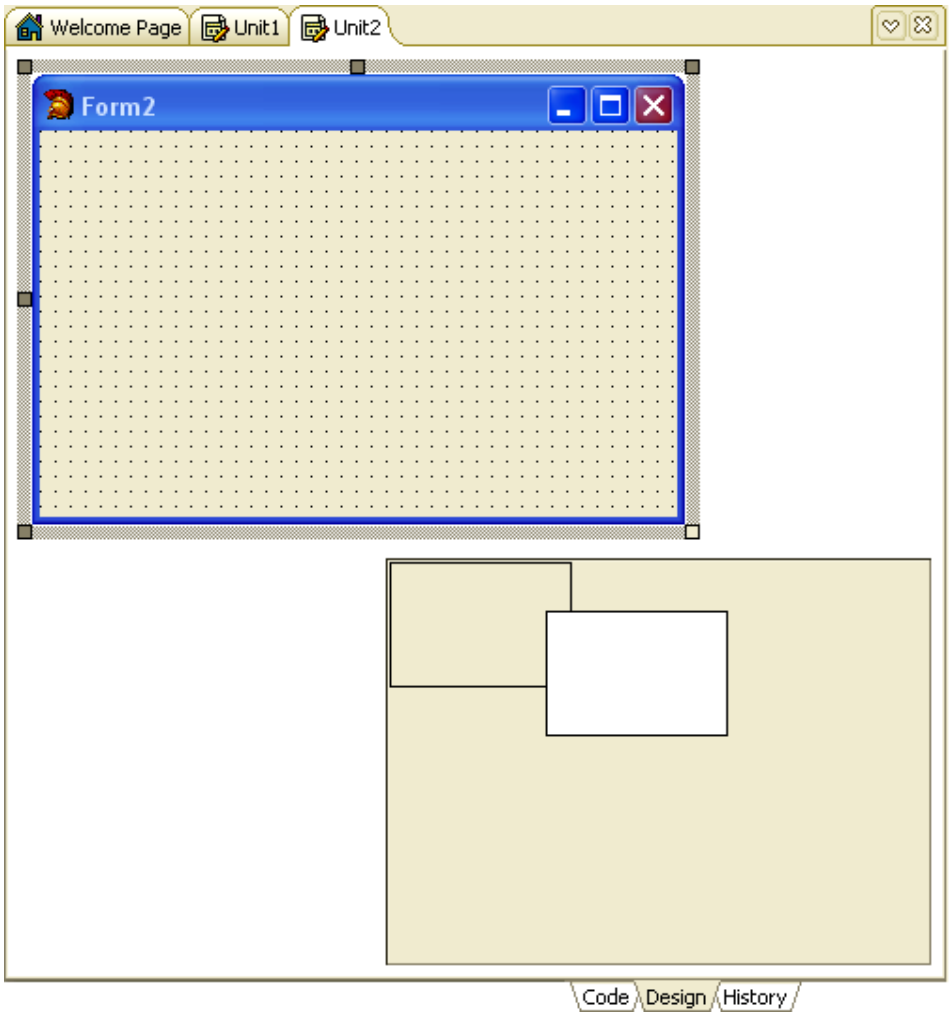


Рис. 2.4 – Зображення екрана в дизайнері форми

Будучи візуальним засобом розміщення компонентів на формі, дизайнер форми дозволяє безпосередньо за допомогою мишки розміщати компоненти на формі з використанням Палітри Інструменту (Tool Palette), Інспектора Об'єктів (Object Inspector) або Дерева Об'єктів (Object Tree). При цьому необхідні зміни у вікні форми можуть бути внесені в будь-який момент під час проектування (наприклад, за допомогою мишки можна змінювати положення і розміри компонентів).

З формою, крім дизайнера форми, пов’язані ще два вікна, розташовані на окремих сторінках разом зі сторінкою дизайнера, а саме: вікно коду (закладка з ім’ям Code) і вікно менеджера історії (закладка з ім’ям History).

2.2.3. Вікно коду

У вікні коду програми відображається і редагується текст програми (найчастіше текст модуля), написаної мовою Delphi. При розробці нового проекту це вікно містить створюваний автоматично мінімальний початковий текст модуля (рис. 1.1). Цей текст забезпечує нормальне функціонування порожньої форми. Далі програміст модифікує його відповідно до розв'язуваної задачі, причому зміни виконуються як за допомогою засобів бібліотеки візуальних компонентів, так і за допомогою «ручної» корекції. При цьому досить часто доводиться переходити з вікна коду у вікно форми і зворотно.

Доступ до вікна коду (рис. 2.5) здійснюється або натисканням клавіші F12, або кліком мишкою над інструментальною кнопкою  на панелі Вигляд ICP, або виконанням команди View ► Toggle Form/Unit. Зворотний перехід у вікно дизайнера форми здійснюється аналогічно. Можна також виконувати кліки мишкою над закладками Code і Design.

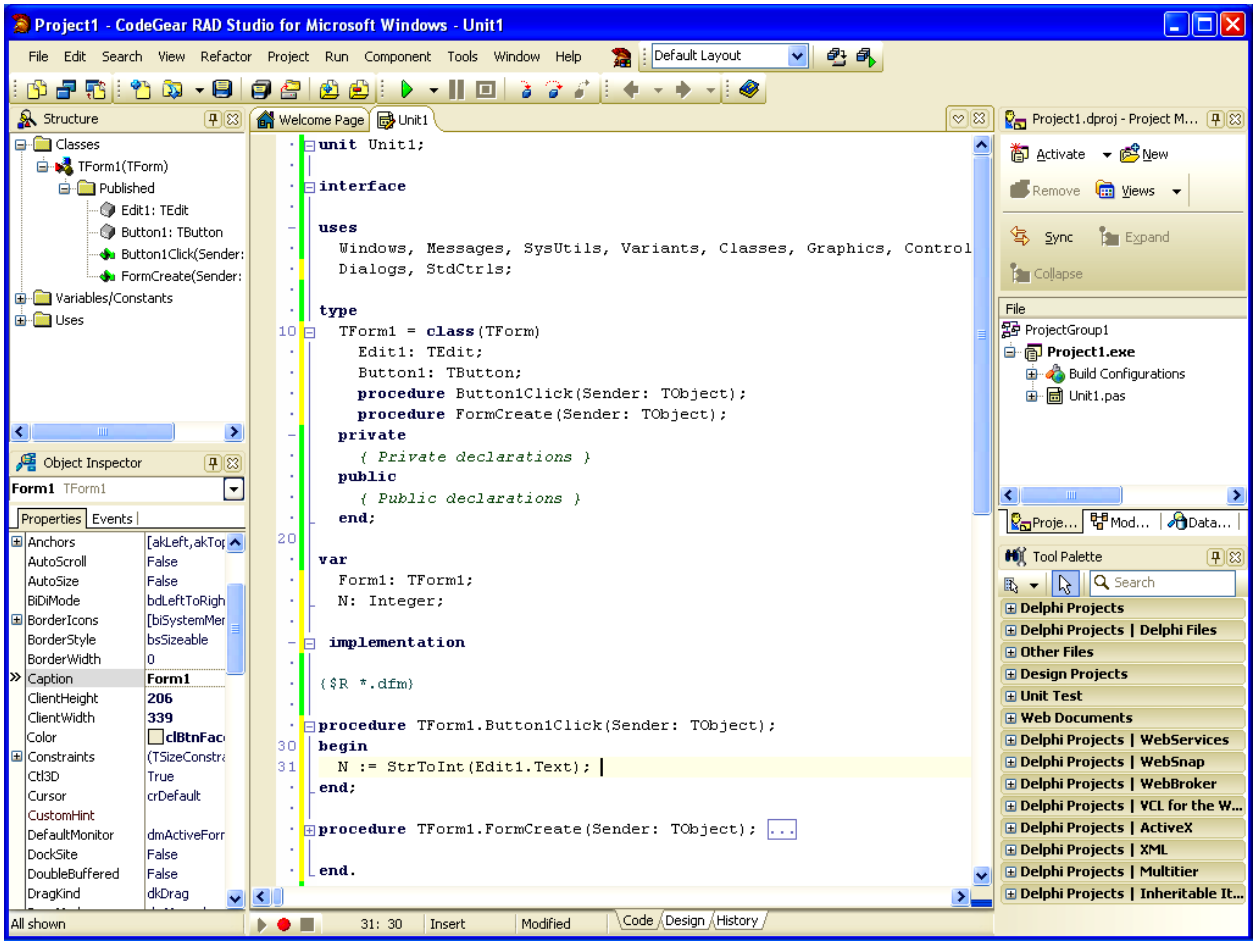


Рис. 2.5 – Вигляд ICP Delphi 2009 з відкритим вікном коду

Вікно коду є звичайним вікном багаторядкового текстового редактора і робота з ним не має ніяких особливостей відносно роботи у будь-якому іншому редакторі:

- набирання тексту здійснюється у позиції, де знаходиться курсор;
- переміщення курсору здійснюється кліком мишкою над потрібним місцем вікна, або за допомогою клавіш керування курсором, а саме:
  - стрілка ліворуч, стрілка праворуч, стрілка вгору, стрілка вниз – переміщення курсору на одну позицію у відповідному напрямі;
  - Home – переміщує курсор на початок поточного рядка;
  - End – переміщує курсор на кінець поточного рядка;
  - Page Up – переміщує курсор на одну сторінку екрану вгору;
  - Page Down – переміщує курсор на одну сторінку екрану вниз;
  - Ctrl+стрілка ліворуч – переміщує курсор на одно слово ліворуч (до найближчого символу-роздільника);
  - Ctrl+стрілка праворуч – переміщує курсор на початок наступного слова (усі символи-роздільники пропускаються);
  - Ctrl+стрілка вгору – переміщує редакторське вікно відносно тексту на один рядок угору (візуально текст у редакторському вікні зсувається на один рядок вниз);
  - Ctrl+стрілка вниз – переміщує редакторське вікно відносно тексту на один рядок вниз (візуально текст у редакторському вікні зсувається на один рядок угору);
  - Ctrl+Home – переміщує курсор на початок тексту;
  - Ctrl+End – переміщує курсор на кінець тексту;
  - Ctrl+Page Up – переміщує курсор на перший рядок тексту у вікні редактора, не змінюючи його положення по горизонталі;
  - Ctrl+Page Down – переміщує курсор на останній рядок тексту у вікні редактора, не змінюючи його положення по горизонталі;
- натискання клавіші Backspace приводить до знищення одного символу безпосередньо перед курсором;
- натискання клавіші Delete приводить до знищення одного символу безпосередньо за курсором;
- клавіша Ctrl+Backspace знищує усі символи поточного слова ліворуч від курсору до найближчого символу-роздільника;

- клавіша Ctrl+Delete знищує усі символи поточного слова праворуч від курсору до найближчого символу-роздільника;
- відновити текст до останнього редагування можна за допомогою пункту меню Edit ►Undo або натисканням клавіші Ctrl+Z (це можна робити декілька разів підряд):
- відновити текст після останнього редагування (у тому числі декілька разів підряд) можна за допомогою пункту меню Edit ►Redo або натисканням клавіші Shift+Ctrl+Z;
- виділення тексту здійснюється зміщенням курсору при натиснутій лівій кнопці мишки або зміщенням курсору за допомогою клавіш керування ним при натиснутій клавіші Shift (можна також натиснути клавішу Shift і клікнути мишкою над позицією, у якій закінчується виділення);
- натискання будь-якої символьної клавіші у разі наявності виділення приводить до заміни виділеного тексту введеним символом;
- натискання клавіш Backspace та Delete приводить до знищення всього виділеного тексту, а не одного символу, як це буває у разі відсутності виділення;
- виділений текст може бути скопійований у буфер обміну, для чого треба вибрати пункт меню Edit ►Copy або натиснути клавішу Ctrl+C (або Ctrl+Insert);
- для копіювання виділеного тексту з його знищенням (вирізання) служить пункт меню Edit ►Cut або клавіша Ctrl+X (або Ctrl+Delete);
- вставка тексту, що знаходиться в буфері обміну, здійснюється вибором пункту меню Edit ►Paste або натисканням клавіші Ctrl+V (або Shift+Insert).

Редактор коду в ICP Delphi 2009 забезпечує нумерацію рядків. За умовчанням пронумеровані всі рядки з номерами, кратними десяти, а також поточний рядок (див. рис. 2.5).

Інтелектуальні можливості редактора коду забезпечують полегшення роботи під час набирання програмного коду. Це забезпечується за допомогою автоматичного формування програмних заготовок для деяких стандартних конструкцій:

- якщо виконується набирання заголовка будь-якого циклу (**`for`**, **`while`**, **`do`**), введення пробілу після першого його слова приводить

до автоматичного формування заготовки для продовження циклу (наприклад для циклу **for** така заготовка має вигляд `for I := 0 to List.Count - 1 do`);

- введення пробілу після слів **if** та **case** приводить до автоматичного формування продовження операторів **if** та **case**;
- якщо після службового слова **begin** натискається клавіша Enter, то автоматично після порожнього рядка формується рядок з закриваючою операторною дужкою **end** і символом «крапка з комою».

Окремі секції коду (модуль, його інтерфейсна секція та секція реалізації, описи класів, процедур, функцій тощо) можуть бути «згорнуті», забезпечуючи приховання згорнутого коду. Для згортання секції слід клацнути мишкою над квадратиком із символом «мінус» поруч із заголовком секції. У цьому випадку секція згортається до свого заголовка, ліворуч від якого у квадратику міститься символ «плюс», а праворуч – прямокутник із трьома крапками. Для «розгортання» згорнутої секції достатньо клацнути мишкою або над квадратиком із символом «плюс», або над прямокутником із трьома крапками. Наприклад, на рис. 2.5 згорнутою є секція опису процедури `TForm1.FormCreate`, а інші секції коду модуля `Unit3` розгорнуті (наприклад, секція **unit** охоплює весь код модуля).

Якщо програма є багатомодульною, кожен модуль може бути завантажений на окремій сторінці вікна коду, причому кожна сторінка має закладку з іменем модуля (на рис. 2.5 завантажений тільки один файл).

2.2.4. Менеджер історії

Менеджер історії (History Manager) розташовується на одній із вкладок вікна дизайнера форми, аналогічно вікну коду. Доступ до нього здійснюється кліком мишкою над закладкою History у нижній частині вікна дизайнера форми. Менеджер історії дозволяє побачити і порівняти різні версії файлу, включаючи резервні копії. Використовуючи менеджер історії, можна, зокрема, повернутися до необхідної версії файлу (рис. 2.6), причому це можна зробити у будь-який час.

2.2.5. Вікно інспектора об’єктів

Вікно Інспектора Об’єктів (Object Inspector) призначене для зміни параметрів (характеристик) розміщених на формі компонентів. За допомогою

мишки у вікні форми можна коректувати тільки частину параметрів, які характеризують компоненти, що розміщені на формі. Користуючись же Інспектором Об'єктів, програміст може коректувати всі характеристики компонентів, які доступні на етапі проектування програми (наприклад, колір, шрифт і текст напису, колір компонента). Для здійснення цих змін в Інспекторі Об'єктів використовуються дві вкладки (сторінки) – Properties (Властивості) та Events (Події). За допомогою вкладки Properties (Властивості) програміст може змінювати властивості компонента, а за допомогою вкладки Events (Події) – призначати реакцію компонента на ту або іншу подію (реакцію на натискання клавіш, клік мишкою, зміну розміру вікна, появу компонента на екрані тощо).

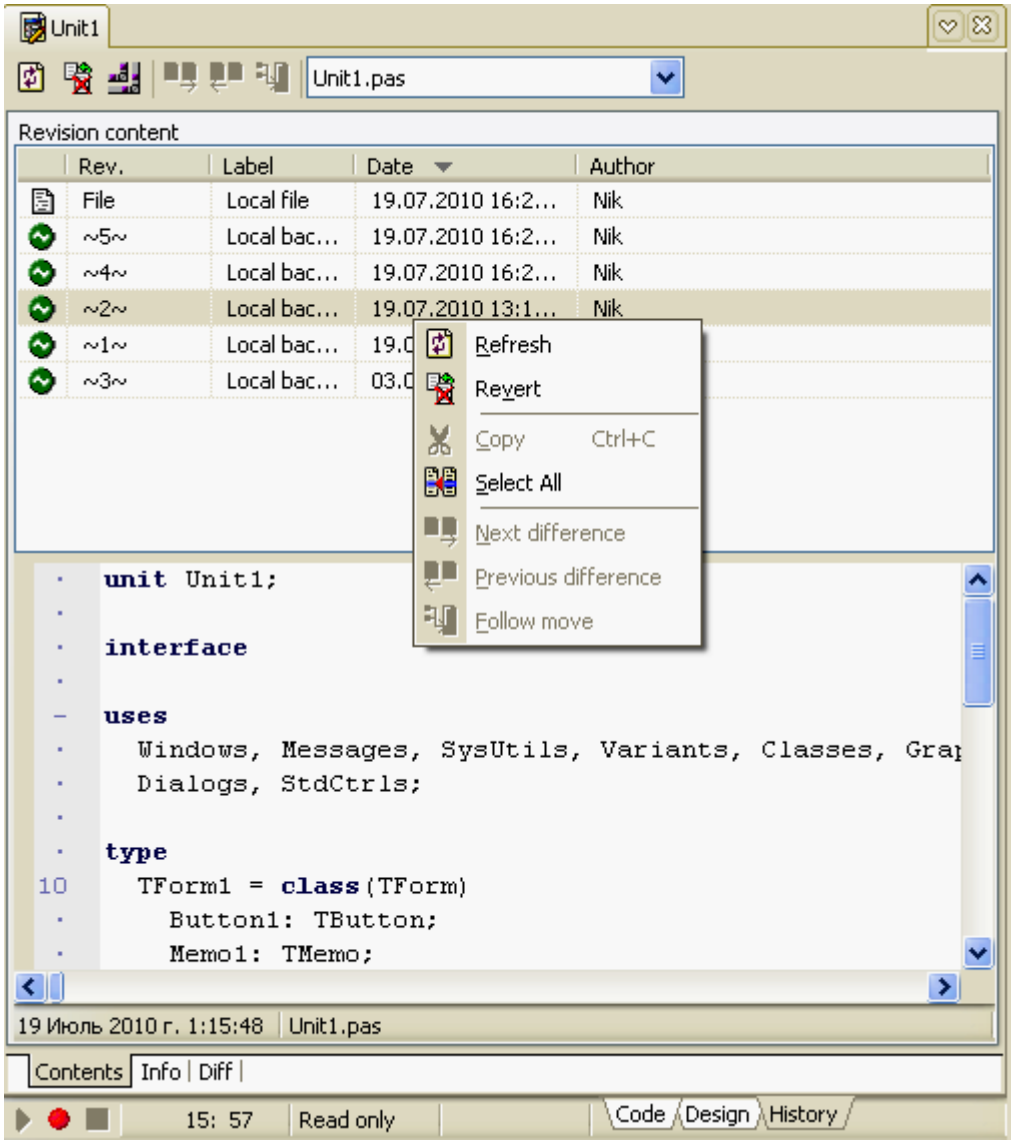


Рис. 2.6 – Сторінка історії змін

Для зміни властивостей компонента необхідно перейти в Інспекторі Об'єктів на першу вкладку, що має вигляд таблиці, у першому стовпчику якої містяться імена властивостей. Значення властивості відображається поруч з її ім'ям у другому стовпчику.

Властивості компонентів можуть відображатися єдиним значенням (числом, рядком символів, True – Істина чи False – Неправда) або множиною значень. У першому випадку говорять про прості властивості, а в другому – про складні. Ліворуч від імені складної властивості відображається значок ⊞.

Для задавання значення простої властивості необхідно вибрати відповідний рядок і ввести потрібне значення в правому стовпчику. Якщо при виборі простої властивості в правому кінці рядка з'являється кнопка ▼, то це означає, що для даної властивості визначений список можливих значень, який розкривається при кліку над кнопкою ▼ і служить для подальшого вибору потрібного значення. Для зміни значення складної властивості слід клацнути мишкою над значком ⊞ поруч з його ім'ям, у результаті чого відкривається список складових цієї властивості. Закриття цього списку здійснюється кліком мишкою над значком ⊟, у який перетворюється значок ⊞ при відкритті списку.

Наприкінці рядка для деяких із властивостей при їх активізації може з'явитися кнопка ⋯. Клік над цією кнопкою приводить до появи на екрані діалогового вікна, що служить для установки значення властивості.

Вкладка Events (Події) також має вигляд таблиці з декількох рядків і двох стовпчиків. У першому стовпчику відображається ім'я події, а в другому – ім'я підпрограми для її опрацювання (ім'я опрацьовувача події).

Крім двох вкладок, у верхній частині вікна Інспектора Об'єктів є список, який може розкриватися і містить імена всіх поміщених на форму компонентів із зазначенням їхніх класів. При кліку мишкою над ім'ям компонента в цьому списку відбувається переключення на відповідні таблиці в Інспекторі Об'єктів, а також активізація обраного компонента у вікні форми, що позначається виділенням компонента прямокутником.

Налаштування вікна Інспектора Об'єктів можна виконати за допомогою контекстного меню, яке відкривається при кліку правою кнопкою мишки.

Повторне відкриття раніше закритого вікна Інспектора Об'єктів виконується командою View ► Object Inspector (клавіша Alt+Enter).

2.2.6. Вікно структури

Вміст вікна структури (Structure) залежить від того, що завантажено в центральне вікно – дизайнер форми або вікно коду. Якщо в центральному вікні відкритий дизайнер форми, то у вікні структури міститься Дерево Об'єктів (Object TreeView), а при відкритому вікні коду у вікно структури завантажується браузер коду.

Дерево Об'єктів наочно відображає зв'язки між окремими візуальними та невізуальними компонентами, розміщеними на активній формі або в активному модулі даних, а саме, приналежність одних компонентів іншим. Окремі компоненти відображаються в Дереві Об'єктів за допомогою піктограм (маленьких рисунків), поруч із якими містяться імена відповідних компонентів. Клацання (клік) лівою кнопкою мишки на кожній з таких піктограм приводить до активізації відповідного компонента у вікні форми і відображення його властивостей у вікні Інспектора Об'єктів. При подвійному кліку на піктограмі компонента спрацьовує так званий механізм Code Insight, що вставляє у вікно коду заготівку для опрацьовувача події OnClick (За кліком) або опрацьовувача іншої події (залежно від типу компонента). При цьому автоматично відкривається вікно коду, а курсор розміщується усередині заготовки опрацьовувача події.

Використовуючи Дерево Об'єктів, можна змінити власника компонента, для чого досить клацнути лівою кнопкою мишки над піктограмою і, не відпускаючи кнопку мишки, «перетягнути» її у вікні на піктограму нового власника. Натискання клавіші Delete приводить до видалення компонента, який виділений у Дереві Об'єктів.

Вікно Браузера Коду (Code Explorer) звичайно активізується разом з вікном коду і служить для полегшення пошуку потрібних елементів у великому тексті програми. Це вікно містить елементи, що використовуються в програмі (рис. 2.7).

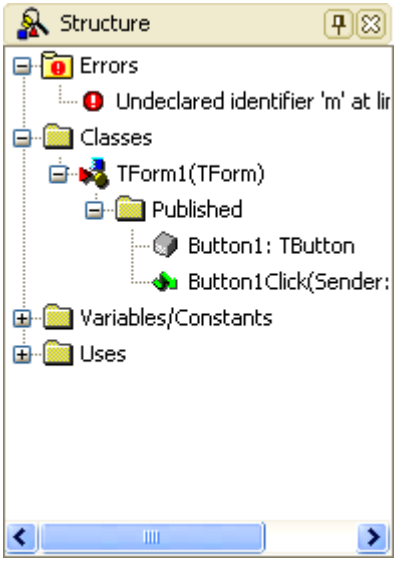


Рис. 2.7 – Браузер коду з помилкою в описі

При подвійному кліку мишкою над елементом у вікні браузера коду текстовий курсор у вікні коду автоматично переміщається на опис цього елемента або на рядок,

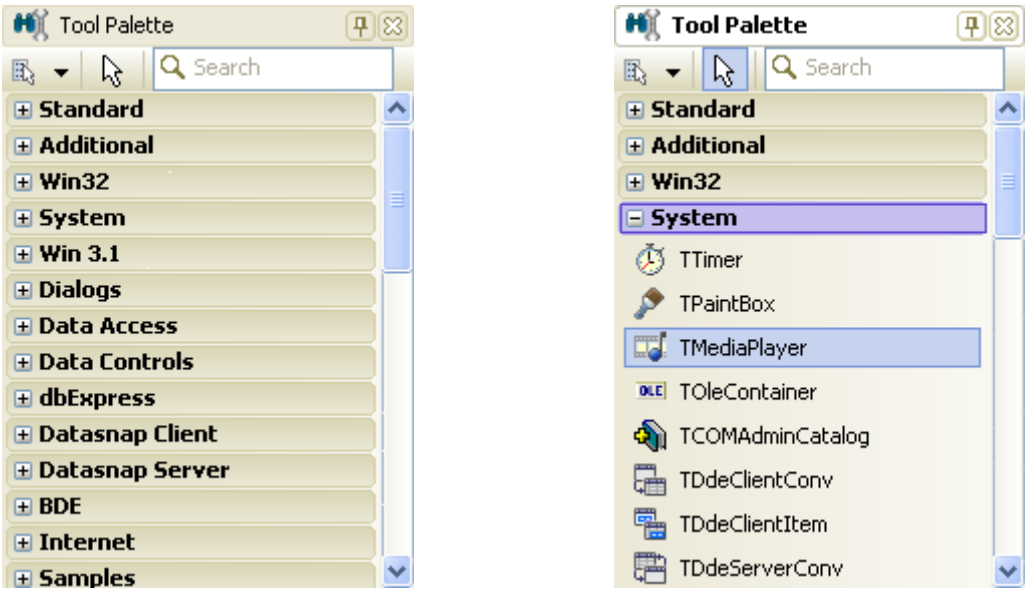
у якому він згадується вперше. За допомогою браузера коду можна також проводити додавання і перейменування елементів програми.

У браузері коду додатково відображаються помилки в описі різних об'єктів, які використовуються в модулі (цей випадок ілюструє рис. 2.7). Кліком мишкою над гілкою з описом помилки можна забезпечити перехід до відповідної позиції у вікні коду.

Закрити вікно структури можна, клікнувши мишкою над кнопкою в його правому верхньому куті. Повторне відкриття раніше закритого вікна структури виконується командою View ► Structure.

2.2.7. Вікно палітри інструменту

Проектування програми у ICP Delphi пратично неможливе без використання палітри інструменту, яка забезпечує додавання компонентів до активної форми при проектуванні інтерфейсу програми. За умовчанням в ICP Delphi 2009 палітра інструменту розміщена в спеціальному вікні Tool Palette, яке розташоване праворуч від вікна дизайнера форми. Окремі вкладки палітри інструменту розміщені у вікні у вигляді списку вкладок, кожна з яких розкривається/закривається кліком мишкою над значком або ліворуч від назви вкладки (рис. 2.8, *а*, *б*).



а *б*

Рис. 2.8 – Вікно палітри інструменту при відкритому вікні форми:
а – з закритими вкладками; *б* – з відкритою вкладкою System

Компоненти при розкритих вкладках відображаються піктограмами, поруч із кожною з яких зазначається клас відповідного компонента (рис. 2.8, *б*). Щоб розмістити компонент на формі, треба вибрати компонент у палітрі інструменту, здійснивши над ним клік мишкою, після чого треба клікнути мишкою над зображенням форми у вікні дизайнера форми. Компоненти можна розміщати на формі не тільки за кліком мишкою, але й перетаскувати з палітри інструменту при натиснутій лівій клавіші мишки.

Істотною особливістю палітри інструменту ICP Delphi 2009 є те, що вона доступна для пошуку з використанням фільтра: можна просто набрати ім’я компонента або частину імені, у результаті чого відкривається список компонентів, імена яких містять як свій початок набраний текст.

2.2.8. Вікно менеджера проектів

Вікно менеджера проектів (Project Manger) розташоване над вікном палітри інструменту і має вигляд, наведений на рис. 2.9. Воно служить для відображення структури проекту, над яким веде роботу розроблювач застосунка. За допомогою менеджера проектів можна додавати до проекту файли (модулі), видаляти їх і перейменовувати. Можна також комбінувати проекти, формуючи зв’язану групу проектів.

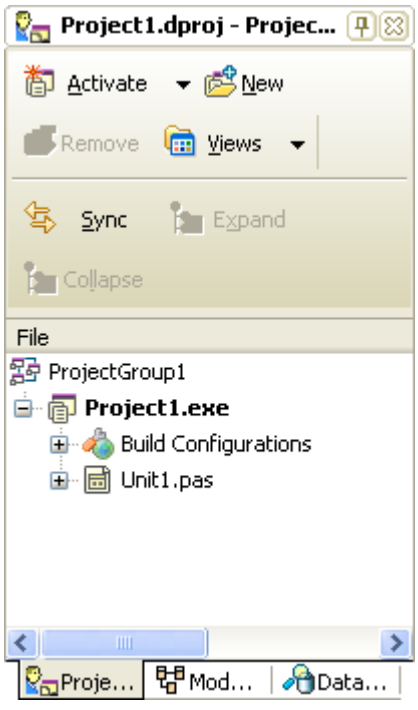


Рис. 2.9 – Вікно менеджера проектів

Власне менеджер проектів завантажується на першу сторінку вікна. На двох інших сторінках розташовані вікно перегляду організації (Model View) і вікно оглядача даних (Data Explorer). Перше з них служить для перегляду логічної структури й організації проекту, а друге – для зміни, видалення, перейменування існуючих і додавання нових зв’язків, а також перегляду бази даних.

У верхній частині вікна менеджера проектів розміщене меню, за допомогою якого можливі корекція проекту та перегляд його структури.

Повторне відкриття раніше закритого вікна менеджера проектів виконується командою View ► Project Manger (Alt+0). Це ж можна забезпечити

виконанням однієї з команд View ► Model View і View ► Data Explorer, які призначені для переходу в менеджері проектів на відповідні вкладки (сторінки).

2.3. Збереження програми та відкриття існуючого проекту

Для збереження вмісту вікна коду треба клікнути мишкою над інструментальною кнопкою  (Save), або обрати пункт меню File ►Save, або натиснути сполучення клавіш Ctrl+S. Якщо раніше збереження коду не здійснювалось, відкриється стандартне вікно збереження файлу, у якому потрібно виконати необхідні дії. У іншому випадку збереження здійсниться без попереджень.

У разі потреби збереження коду у файлі з новим іменем відносно того, з яким він записувався на диск раніше, треба вибрати пункт меню File ►Save As..., результатом чого буде відкриття стандартного вікна збереження файлу.

У Delphi 2009 модуль функціонує не самостійно, а у складі проекту, який створюється автоматично. Збереження проекту разом з текстами підключених до нього модулів здійснюється обиранням пункту меню File ►Save All або кліком мишкою над інструментальною кнопкою  (можна також натиснути клавішу Shift+Ctrl+S). Проект може бути збережений також після обирання пункту меню File ►Save Project As...

Для відкриття існуючого пректу або коду модуля використовується інструментальна кнопка  . Можна також виконати команду File ►Open.

2.4. Компіляція та компонування

Компілятор (Compiler) – це особлива програма, для якої як вхідні дані виступають програми, написані мовою високого рівня, і на виході якої також утворюються програми, але вже написані машинною мовою.

Програми, отримані на виході компілятора, називаються ***об'єктними програмами***, або ***об'єктними кодами***. Особливістю таких програм є те, що вони не можуть бути відразу виконані, оскільки до того необхідно здійснити підключення до них деяких службових підпрограм.

Остаточне опрацювання програми, точніше об'єктного коду, здійснює так звана ***програма-компонувальник*** (інакше ***редактор зв'язків***, Linker). Саме компонувальник створює ***машинний код***, який і виконується комп'юте-

ром при розв'язанні задачі. Таким чином, компонувальник поєднує об'єктний код програми з машинними кодами службових підпрограм, які використовуються даною програмою.

У Delphi 2009 компілятор записує об'єктний код модуля у файл з розширенням .DCU. Якщо компонувальник завершує роботу без помилок, отриманий машинний код записується у так званий exe-файл (він має розширення .EXE), який і містить виконувану програму.

3. ПРОЕКТУВАННЯ НАЙПРОСТІШОГО ЗАСТОСУНКА

3.1. Проектування програми

Спроектуємо програму для розв'язання такої задачі

```
//Ввести число. Вважаючи його послідовно довжиною сторони
//квадрата, радіусом кола або кулі, обчислити і вивести
//на екран 1) площу квадрата, 2) площу круга й 3) об'єм
//кулі.
```

Завантаживши Delphi, виберемо команду File ►New ►Other... ►VCL Forms Application, у результаті чого буде створена порожня форма. Захоплюючи по черзі межі форми при натиснутій лівій кнопці мишки, підберемо зручні розміри форми. Далі слід розмістити на формі компоненти відповідно до вимог задачі.

Виберемо на вкладці Standard палітри інструменту компонент TPanel (Панель) і клікнемо мишкою над формою. У результаті на формі з'явиться компонент TPanel (лівий верхній кут компонента буде розташовуватися в точці, над якою здійснювався клік). Перетягнемо панель мишкою в нижню частину форми, розтягши її на всю ширину форми і задавши висоту, що дорівнює приблизно 0,2 від висоти форми (панель можна розташовувати в будь-якому місці форми). Вона буде служити контейнером, у якому розміститься інші компоненти проекту (насамперед вікна редакторів для введення даних і кнопки для подачі команди на початок обчислень та команди на завершення роботи програми). У принципі без панелі можна обійтися, розмістивши згадані елементи на вільних місцях форми, що є контейнером для всіх компонентів.

Аналогічним способом виберемо на вкладці Standard компоненти TLabel, TEdit та TButton, а на вкладці Additional – компонент TBitBtn. Розмістимо ці компоненти на панелі (для чого будемо клікати мишкою над панеллю) і встановимо для них потрібні розміри і місце розташування.

Компонент TLabel (Мітка) призначений для виведення різних повідомлень (підказок, результатів обчислень і т. д.); ми будемо його використовувати для виведення підказок, що випереджають введення. Для зміни тексту, що міститься в цьому компоненті, досить змінити значення його властивості Caption, що можна робити як на етапі проектування програми, так і програмними засобами.

Компонент TEdit (однорядкове редаговане текстове поле) застосовують для введення та (або) відображення досить довгих текстових рядків. Для зміни тексту, що міститься в ньому, достатньо змінити значення його властивості Text, що також можна виконувати як на етапі проектування програми, так і програмними засобами. Ми будемо використовувати цей компонент для введення числа.

Компонент TButton (Кнопка) призначений для керування програмою. Ми вдамося до нього для організації обчислень і забезпечення виведення.

Компонент TBitBtn (Кнопка з зображенням) є різновидом кнопки Button. Ми застосуємо один з варіантів цього компонента, який служить для видачі сигналу про припинення роботи програми. Ця кнопка не є обов'язковою, оскільки припинення роботи програми забезпечується кліком над кнопкою закриття форми або натисканням сполучення клавіш Alt+F4. Замість цього компонента можна використовувати і кнопку TButton.

Нарешті, для виведення результату обчислень помістимо на форму багаторядкове редаговане текстове поле, для чого виберемо на вкладці Standard компонент TMemo (Текстове поле, від *memo field*) і клацнемо мишкою над вільним місцем форми (поза панеллю). Компонент TMemo вживають для введення, редагування й (або) відображення досить довгих текстів, які можуть містити кілька рядків. Цей текст може корегуватися як на етапі візуального проектування, так і програмно, для чого необхідно змінювати властивість Lines (Рядки) цього компонента.

У результаті початкового проектування програми вікно форми може набути, наприклад, вигляду, наведеного на рис. 3.1.

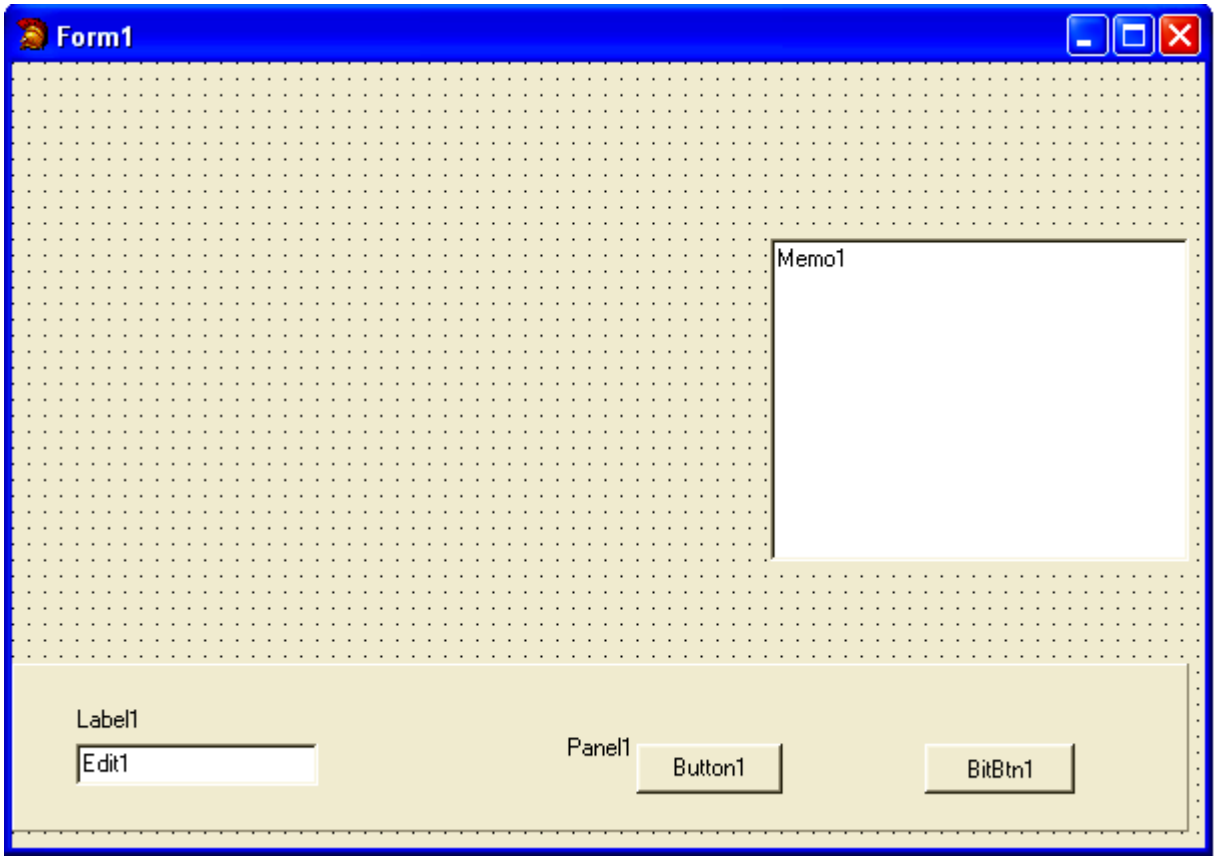


Рис. 3.1 – Можливий початковий вигляд форми

Відповідно до специфіки розв’язуваної задачі здійснимо такі зміни властивостей компонентів:

- Форма:
  Position — poScreenCenter
  Caption — Приклад
- Панель:
  Align — alBottom
  BevelOuter — bvNone
  Caption — очистимо
- Мітка:
  Caption — Уведіть число і натисніть "Обчислити"
  Name — lbOutput1
- Поле введення:
  Name — edInput1
  Text — 0
- Багаторядкове поле:
  Align — alClient

Lines — очистимо
Name — mmOutput1
ScrollBars — ssBoth

- Кнопка Button1:
  Caption — Обчислити
- Кнопка Close:
  Kind — bkClose
  Name — bbClose

У результаті форма набуде вигляду, зображеного на рис. 3.2.

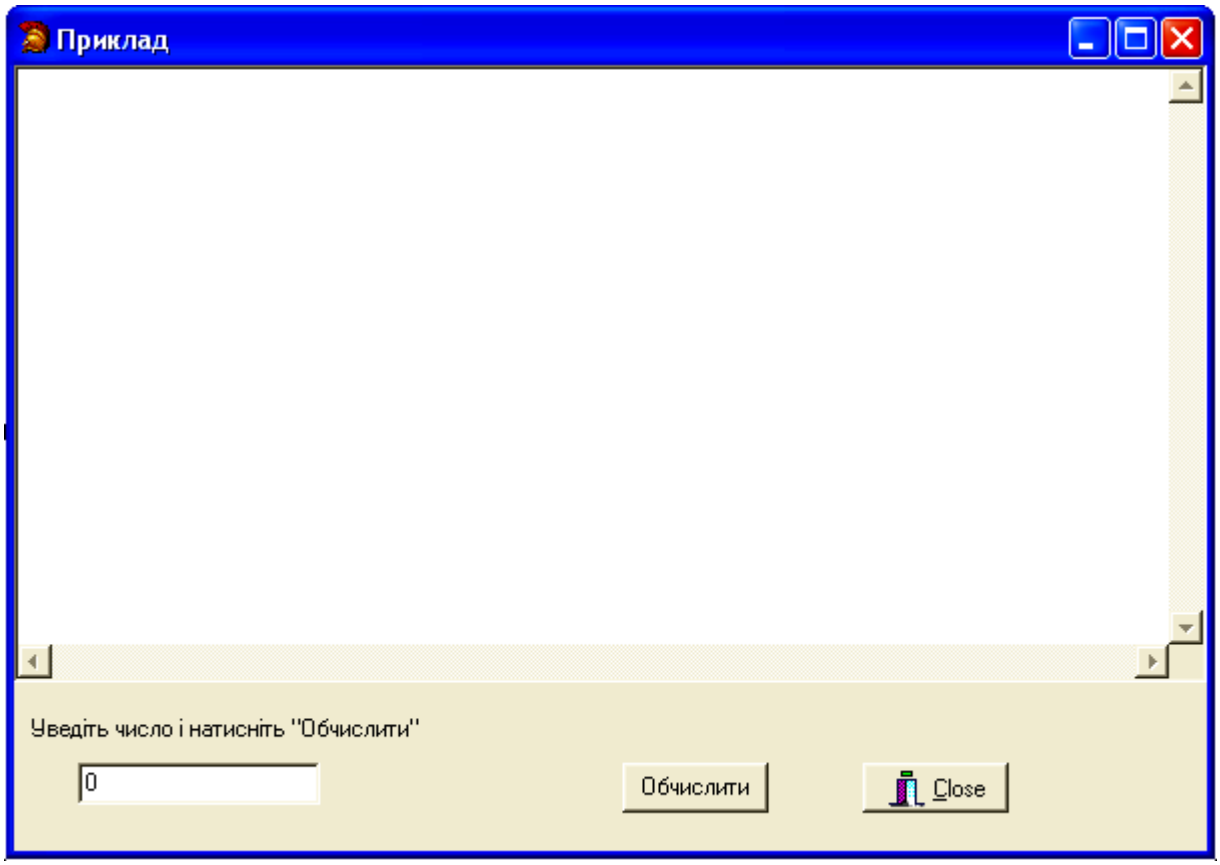


Рис. 3.2 – Можливий остаточний вигляд форми

Розглянемо описані вище зміни.

Властивість Position (Положення) у форми визначає її положення щодо меж екрана. За умовчанням ця властивість має значення poDesigned й означає розташування вікна так, як воно було розміщене на етапі конструювання. Наведене значення poScreenCenter для властивості Position зумовлює розташування форми в центрі екрана.

Властивість Align (Вирівнювання) визначає спосіб розміщення даного компонента щодо того контейнера, який його містить. Наприклад, панель розташована усередині форми, а кнопки, мітка і поле введення – усередині панелі. Значення alBottom у цієї властивості, задане для панелі, забезпечує притиснення останньої до нижньої межі форми з розтягуванням її по всій довжині форми. У багаторядкового поля властивість Align дістала значення alClient, що «примушує» компонент зайняти всю частину форми, яка залишилася незаповненою (вона називається клієнтською областю).

Оскільки в компонента може бути зовнішня крайка, її досить часто усувають, щоб компонент не виділявся на фоні свого контейнера. Зміна вигляду зовнішньої крайки забезпечується зміною значення властивості BevelOuter. У панелі їй надане значення bvNone – немає крайки.

Властивість Name (Ім'я) визначає ім'я, під яким компонент буде існувати в програмі. У більшості компонентів ця Властивість змінена з метою додавання імені значеннєвого навантаження (lbOutput1 – мітка (Label) виведення, edInput1 – редактор (Edit) введення, mmOutput1 – поле (Memo) виведення, bbClose – бітова кнопка (BitBtn) закриття).

Властивість Caption (Заголовок) є в багатьох компонентів Delphi і служить для виведення заголовка. Вона змінена у трьох компонентів: у форми – для виведення тексту Приклад, у панелі – для знищення заголовка, у мітки – для виведення підказки .

Властивість Text (Текст) у поля введення і властивість Lines (Рядки) у багаторядкового поля визначають текст, що буде міститися в однорядковому або багаторядковому полі в момент їхньої появи на екрані. Для компонента TEdit у властивість Text записане значення 0, щоб навіть за відсутності введеного числа за нього було взяте значення 0. У багаторядкового поля властивість Lines очищена.

Значення ssBoth (Обидві) властивості ScrollBars у компонента TMemo забезпечить розміщення в цьому компоненті двох смуг прокручування (скролінга) з метою можливого перегляду текстів, що вийшли за межі компонента.

Значення bkClose властивості Kind (Сорт) у компонента TBitBtn означає наявність на поверхні кнопки типового значка і напису Close, а також зв'язок з кнопкою функції закриття вікна.

Зміна значень властивостей буде автоматично приводити до зміни тексту модуля. В результаті текст модуля прийме наступний вид:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants,
Classes,Graphics,
  Controls, Forms, Dialogs, StdCtrls, Buttons, ExtCtrls;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    lbOutput1: TLabel;
    edInput1: TEdit;
    Button1: TButton;
    bbClose: TBitBtn;
    mmOutput1: TMemo;
    private
      { Private declarations }
    public
      { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Як це видно з наведеного вище тексту модуля, Delphi автоматично вставив у клас розміщені на формі компоненти, надавши їм імена або стандартні (якщо компонент не перейменовувався при проектуванні форми) – `Panel1` та `Button1`, або ті, які були дані їм програмістом (`lbOutput1`, `edInput1`, `bbClose`, `mmOutput1`). Перелік вставлених нами компонентів і становить більшу частину опису класу. Для завершення проектування програми залишилося доповнити її елементами, що власне і забезпечують орга-

нізацію обчислень і виведення даних. Тут також багато в чому допомагає ICP.

Виконаємо подвійний клік мишкою над зображенням форми в панелі Дерево Об'єктів (можна також клікнути над будь-яким вільним місцем форми в її вікні, але зараз на формі вільне місце відсутнє). У відповідь на клік мишкою Delphi автоматично вставить у код модуля перед термінатором **end** з крапкою такий код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin

end;
```

Це код так званого опрацьовувача події OnCreate (За створенням) для форми.

Слово **procedure** вказує, що опрацьовувач є так званою підпрограмою-процедурою. Складене ім'я `TForm1.FormCreate` – це ім'я процедури, що складається з двох частин: імені класу `TForm1` і власне імені процедури `FormCreate`. Після імені процедури в круглих дужках зазначено опис параметра, з яким вона буде викликатися (`Sender: TObject`). У нашій програмі параметр `Sender` використовуватися не буде. У більш складних програмах за допомогою цього параметра програміст може визначити, який компонент згенерував дану подію (у даному випадку подія OnCreate). У загальному випадку у процедури може бути кілька параметрів; вони можуть також бути відсутні. Перший рядок розглянутого фрагмента коду називається ***заголовком процедури*** (він завершується символом «крапка з комою»).

Слідом за заголовком процедури розташовується її ***тіло*** (у даному випадку воно порожнє й містить тільки так звані операторні дужки **begin** та **end**). Щоб опрацьовувач події виконував які-небудь дії, його тіло повинне бути наповнене операторами. Відзначимо, що відразу ж за заголовком може розміщуватися секція описів, у якій програміст описує допоміжні елементи, необхідні для реалізації тіла процедури.

Крім вставки коду опрацьовувача події, Delphi автоматично модифікує опис класу, вставляючи в нього перед словом **private** заголовок опрацьовувача події OnCreate форми:

```
procedure FormCreate (Sender: TObject);
```

Модифікуємо код опрацьовувача події OnCreate форми, надавши йому такого вигляду:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   Caption := 'Приклад для самостійної роботи';
   DecimalSeparator := '.';
end;
```

Перший оператор цієї процедури змінює текст заголовка форми на текст Приклад для самостійної роботи. Цей текст буде виведений у верхній частині форми. Про другий оператор потрібно поговорити особливо. Справа в тому, що якщо використовується русифікована версія Windows, то в ній як роздільник цілої та дробової частин дійсного числа використовується не десяткова точка, як це має місце в мовах програмування (у тому числі і в Delphi), а кома. Результатом цього може бути неправильна робота деяких підпрограм, що перетворюють рядки до дійсних чисел (наприклад, StrToFloat). Системна змінна DecimalSeparator містить символ, що використовується як роздільник цілої та дробової частин дійсних чисел. Другий оператор процедури TForm1.FormCreate на початку роботи програми змінює цей символ на символ «крапка», що використовується в Delphi та у не русифікованій версії Windows.

Клікнемо тепер мишкою над кнопкою Обчислити. У відповідь на клік мишкою Delphi автоматично вставить у код модуля перед термінатором **end** з крапкою такий код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Це код так званого опрацьовувача події OnClick (За кліком) для кнопки Обчислити. Взагалі кажучи, при кліку мишкою в працюючій програмі виникає подія OnClick, що зв'язується з компонентом, над яким був здійснений клік.

Якщо опрацьовувач події відсутній, то подія ніяк не опрацьовується; за умови його наявності клік мишкою активізує опрацьовувач і забезпечує виконання дій, записаних у цьому опрацьовувачі.

Крім того, в опис класу перед словом **private** буде автоматично вставлений заголовок опрацьовувача події OnClick кнопки Button1:

```
procedure Button1Click(Sender: TObject);
```

Модифікуємо код процедури `TForm1.Button1Click`, надавши йому такого вигляду:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  X := StrToFloat(edInput1.Text);
  mmOutput1.Lines.Add('X=' + FloatToStr(X));
  mmOutput1.Lines.Add('Площа квадрата дорівнює '
                      + FloatToStr(Sqr(X)));
  mmOutput1.Lines.Add('Площа круга дорівнює '
                      + FloatToStr(Pi * Sqr(X)));
  mmOutput1.Lines.Add('Об''єм кулі дорівнює '
                      + FloatToStr(Pi * X * X * X / 4));
  lbOutput1.Caption := 'Уведіть число' +
    ' і натисніть "Обчислити" або припиніть обчислення';
  edInput1.SetFocus;
end;
```

Додамо також в інтерфейсну частину модуля після рядка

```
Form1: TForm1;
```

рядок

```
X: Real;
```

Цей рядок описує змінну з ім’ям `X`, що служить для зберігання значень реального (дійсного) типу. Ця змінна буде використовуватися у програмі для зберігання числа, що вводиться. Подібного роду опис дозволяє надалі використовувати ім’я `X` усюди в модулі нижче точки оголошення з можливістю зміни значення змінної, позначеної цим ім’ям.

Розглянемо призначення операторів, що утворюють тіло процедури `TForm1.Button1Click`.

Оператор

```
X := StrToFloat(edInput1.Text);
```

забезпечує перетворення тексту, що міститься у властивості `Text` поля введення, до дійсного числа і запис отриманого значення в змінну `X`.

Наступний оператор виводить у поточний рядок багаторядкового редактора `mmOutput1` текст `X=` та перетворене до рядкового вигляду значення змінної `X`.

Третій оператор аналогічним способом виводить текст `Площа квадрата дорівнює` та перетворений до рядкового виду результат обчислення площі квадрата.

Далі виводиться текст `Площа круга дорівнює` та перетворений до рядкового виду результат обчислення площі круга, а також текст `Об'єм кулі дорівнює` та перетворений до рядкового вигляду результат обчислення об'єму кулі. Зазначимо, що для запису в рядок символу «апостроф» потрібно записати два апострофи підряд (безпосередньо в рядок при цьому буде записаний один апостроф).

Наголосимо, що оскільки дані вводяться в рядковому (а не числовому!) виді, введене значення необхідно привести до дійсного числа, для чого в Delphi визначена функція `StrToFloat`. Аналогічно, оскільки в Delphi можливе виведення тільки текстових повідомлень, числові значення повинні бути перетворені до так званого рядкового виду (для подібного роду перетворення дійсного числа в Delphi визначена функція `FloatToStr`). В обчисленнях використана визначена в Delphi функція піднесення у квадрат дійсного значення (`Sqr`), а також функція без параметрів `Pi`, що служить для обчислення числа π.

Передостанній оператор змінює властивість `Caption` мітки `lbOutput1` з метою зміни підказки для введення значень.

Останній оператор тіла опрацьовувача `Button1Click` за допомогою методу `SetFocus` передає контроль над клавіатурою полю введення `edInput1` з метою забезпечення можливості введення нового числа без додаткових дій, якщо необхідно повторити обчислення без виходу з програми і її повторного запуску.

3.2. Остаточний вміст модуля форми

Остаточний текст модуля набуває такого вигляду:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons,
  ExtCtrls;
```

```
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    lbOutput1: TLabel;
    edInput1: TEdit;
    Button1: TButton;
    bbClose: TBitBtn;
    mmOutput1: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    private
      { Private declarations }
    public
      { Public declarations }
  end;

var
  Form1: TForm1;
  X: Real;
implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Caption := 'Приклад для самостійної роботи';
  DecimalSeparator := '.';
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  X := StrToFloat(edInput1.Text);
  mmOutput1.Lines.Add('X=' + FloatToStr(X));
  mmOutput1.Lines.Add('Площа квадрата дорівнює '
                      + FloatToStr(Sqr(X)));
  mmOutput1.Lines.Add('Площа круга дорівнює '
                      + FloatToStr(Pi  * Sqr(X)));
  mmOutput1.Lines.Add('Об''єм кулі дорівнює '
                      + FloatToStr(Pi * X * X * X / 4));
  lbOutput1.Caption := 'Уведіть число' +
    ' і натисніть "Обчислити" або припиніть обчислення';
  edInput1.SetFocus;
end;

end.
```

4. ЗАПИТАННЯ ДЛЯ КОНТРОЛЮ ТА САМОКОНТРОЛЮ

1. Перелічте складові класу.
2. Опишіть структуру проекту.
3. Опишіть структуру модуля.
4. Які дії треба виконати для створення нового застосунка?
5. Яке призначення основних вікон інтегрованого середовища Delphi.
6. Яке призначення вікна дизайнера форми?
7. Яке призначення вікна коду?
8. Яке призначення вікна дизайнера форми?
9. Яке призначення менеджера історії?
10. Яке призначення вікна інспектора об’єктів?
11. Яке призначення вікна палітри інструменту?
12. Яке призначення вікна інспектора об’єктів?
13. Опишіть дії, пов’язані з керуванням курсором у вікні редактора.
14. Як можна здійснювати видалення символів у тексті редакторського вікна? Опишіть дію клавіш видалення.
15. Як здійснюється виділення тексту?
16. Як можна скопіювати фрагмент тексту в інше місце?
17. Як можна перенести фрагмент тексту в інше місце?
18. Як здійснюється збереження вмісту вікна коду?
19. Як здійснюється збереження вмісту вікна коду у файлі з новим іменем?
20. Як записати на диск проект разом із файлом коду модуля?
21. Яке призначення компілятора?
22. Що створює компілятор у разі відсутності синтаксичних помилок?
23. Яке призначення компонувальника (редактора зв’язків) і що є результатом його роботи?

5. ЗАВДАННЯ НА СОМОСТІЙНУ РОБОТУ

1. Ознайомитись з інтегрованим середовищем Delphi 2009.
2. Повторити дії розділу 3 з проектування найпростішого застосунка.
3. Декілька разів запустити програму на виконання та впевнитись у її працездатності при коретктниз початкових даних.

СПИСОК ЛІТЕРАТУРИ

1. Безменов, М. І. Основи програмування в середовищі Delphi : навч. посіб. / М. І. Безменов. – Х. : НТУ «ХПІ», 2010. – 608 с.
2. Архангельский, А. Я. Программирование в Delphi 6 / А. Я. Архангельский. – М. : БИНОМ, 2002. – 1120 с.
3. Культин, Н. Б. Основы программирования в Delphi 2007 / Н. Б. Культин. – СПб. : БХВ-Петербург, 2008. – 480 с.

ЗМІСТ

Навчальне видання

Методичні вказівки
до самостійної роботи
«Створення та налаштування програм у візуальному середовищі Delphi 2009»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
(спеціалізація «Соціальна інформатика»)

Укладач БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горелий

За авторською редакцією

План 2011 р., поз. 10 / 70-11

Підп. до друку 23.05.2011 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 2,3. Наклад 50 прим.
Зам. № 187. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21